Axiomatische Semantik von WHILE

In der Folge: ...Korrektheit und Vollständigkeit der axiomatischen Semantik

Erinnerung:

- *Hoare-Tripel* (syntaktische Sicht) bzw. *Korrektheitsformeln* (semantische Sicht) der Form

$$\{p\}\ \pi\ \{q\}$$

- Gültigkeit einer Korrektheitsformel im Sinne
  - *partieller* Korrektheit
  - *totaler* Korrektheit

Partielle und totale Korrektheit

- Die Zustandsmenge

$$Ch(p) =_{df} \{\sigma \in \Sigma \mid [\![\, p \,]\!]_B(\sigma) = \mathtt{tt}\}$$

heißt *Charakterisierung von* $p \in$ **Bexp**.

- *Semantik von Korrektheitsformeln:*
  Eine Korrektheitsformel $\{p\}\ \pi\ \{q\}$ heißt
  - *partiell korrekt* (in Zeichen: $\models_{pk} \{p\}\ \pi\ \{q\}$), falls $[\![\, \pi \,]\!](Ch(p)) \subseteq Ch(q)$
  - *total korrekt* (in Zeichen: $\models_{tk} \{p\}\ \pi\ \{q\}$), falls $\{p\}\ \pi\ \{q\}$ partiell korrekt ist und $Def([\![\, \pi \,]\!]) \supseteq Ch(p)$ gilt.

  Konvention: $[\![\, \pi \,]\!](Ch(p)) =_{df} \{[\![\, \pi \,]\!](\sigma) \mid \sigma \in Ch(p)\}$

Hoare-Kalkül HK_{PK} für partielle Korrektheit

[skip] $$\frac{\text{—}}{\{p\}\ skip\ \{p\}}$$

[abort] $$\frac{\text{—}}{\{p\}\ abort\ \{q\}}$$

[ass] $$\frac{\text{—}}{\{p[t\backslash x]\}\ x{:=}t\ \{p\}}$$

[comp] $$\frac{\{p\}\ \pi_1\ \{r\},\ \{r\}\ \pi_2\ \{q\}}{\{p\}\ \pi_1;\pi_2\ \{q\}}$$

[ite] $$\frac{\{p \wedge b\}\ \pi_1\ \{q\},\ \{p \wedge \neg b\}\ \pi_2\ \{q\}}{\{p\}\ \text{if } b \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi}\ \{q\}}$$

[while] $$\frac{\{I \wedge b\}\ \pi\ \{I\}}{\{I\}\ \text{while } b \text{ do } \pi \text{ od}\ \{I \wedge \neg b\}}$$

[cons] $$\frac{p \Rightarrow p_1,\ \{p_1\}\ \pi\ \{q_1\},\ q_1 \Rightarrow q}{\{p\}\ \pi\ \{q\}}$$

Hoare-Kalkül HK_{TK} für totale Korrektheit

...identisch mit HK_{PK}, wobei aber Regel [while] ersetzt ist durch:

[while$_{TK}$] $$\frac{I \wedge b \Rightarrow u[t/v],\ \{I \wedge b \wedge t{=}w\}\ \pi\ \{I \wedge t{<}w\}}{\{I\}\ \text{while } b \text{ do } \pi \text{ od}\ \{I \wedge \neg b\}}$$

wobei

- u Boolescher Ausdruck über der Variablen v; t Term,
- w Variable, die in I, b, π und t nicht frei vorkommt,
- $M =_{df} \{\sigma(v) \mid \sigma \in \Sigma\ \wedge\ [\![\, u \,]\!]_B(\sigma) = \mathtt{tt}\}$ noethersch geordnete Menge (sog. noethersche Halbordnung).

Korrektheit und Vollständigkeit von HK_{PK} und HK_{TK}

Sei K ein Kalkül für partielle bzw. totale Korrektheit

Zentral sind dann die Fragen der...

- *Korrektheit:* ...ist jede mithilfe von K ableitbare Korrektheitsformel partiell bzw. total korrekt?
- *Vollständigkeit:* ...ist jede partiell bzw. total korrekte Korrektheitsformel mithilfe von K ableitbar?

Speziell:

- Sind HK_{PK} und HK_{TK} *korrekt* und *vollständig*?

Hauptresultate

Zur Korrektheit:

Theorem [Korrektheit von HK_{PK} und HK_{TK}]

1. HK_{PK} ist korrekt, d.h. jede mit HK_{PK} ableitbare Korrektheitsformel ist gültig im Sinne partieller Korrektheit.
2. HK_{TK} ist korrekt, d.h. jede mit HK_{TK} ableitbare Korrektheitsformel ist gültig im Sinne totaler Korrektheit.

Beweis ...durch Induktion über die Anzahl der Regelanwendungen im Beweisbaum zur Ableitung der Korrektheitsformel.

Zur Vollständigkeit:

Für Korrektheitskalküle ist i.a. nur sog. *relative* Vollständigkeit beweisbar. Das gilt auch für HK_{PK} und HK_{TK}. Details dazu später.

Beispiele 1(2)

...Beweis partieller Korrektheit von Hoareschen Zusicherungen anhand zweier Programme zur Berechnung

- der Fakultät und
- der ganzzahligen Division mit Rest

Beispiele 2(2)

Im Detail:

Beweise, dass die beiden Hoareschen Zusicherungen

$$\{a > 0\}$$
$$x := a;\ y := 1;\ \text{while } x > 1 \text{ do } y := y * x; x := x - 1 \text{ od}$$
$$\{y = a!\}$$

und

$$\{x \geq 0 \wedge y > 0\}$$
$$q := 0; r := x; \text{while } r \geq y \text{ do } q := q + 1; r := r - y \text{ od}$$
$$\{x = q * y + r \wedge 0 \leq r \leq y\}$$

gültig sind im Sinne partieller Korrektheit.

In der Folge geben wir die Beweise dafür in baumartiger Notation an...

Bew. part. Korrektheit: Fakultät (1)

Erster Beweis

```
                                    ____                                   ____
                   ---------------------------------- Ass  ------------------------------ Ass
                   {(y*x)*(x-1)!=a! & (x-1)>0} y:=y*x {y*(x-1)!=a! & (x-1)>0}   {y*(x-1)!=a! & (x-1)>0} x:=x-1 {y*x!=a! & x>0}
                   ------------------------------------------------------------------------------- Comp
y*x!=a! & x>0 & x>1 ==> (y*x)*(x-1)!=a! & (x-1)>0   {(y*x)*(x-1)!=a! & (x-1)>0} y:=y*x; x:=x-1 {y*x!=a! & x>0}
---------------------------------------------------------------------------------------- Cons
                   {y*x!=a! & x>0 & x>1} y:=y*x; x:=x-1 {y*x!=a! & x>0}
                   --------------------------------------------------- While
    T              {y*x!=a! & x>0} WHILE x>1 DO y:=y*x; x:= x-1 OD {y*x!=a! & x>0 & ~(x>1)}   y*x!=a! & x>0 & ~(x>1) ==> y=a!
------------ Cons  ------------------------------------------------------------------------------------------------ Cons
{a>0} x:=a; y:=1 {y*x!=a! & x>0}   {y*x!=a! & x>0} WHILE x>1 DO y:=y*x; x:= x-1 OD {y=a!}
--------------------------------------------------------------------------------------------- Comp
            {a>0} x:=a; y:=1; WHILE x>1 DO y:=y*x; x:=x-1 OD {y=a!}
```

wobei

```
                 ____                                            ____
      ---------------------------------------- Ass  ------------------------------------------ Ass
      {1*a!=a! & a>0 & a=a} x:=a {1*x!=a! & a>0 & x=a}   {1*x!=a! & a>0 & x=a} y:=1 {y*x!=a! & a>0 & x=a}
T ≡ { ------------------------------------------------------------------------------------- Comp
      a>0 ==> 1*a!=a! & a>0 & a=a   {1*a!=a! & a>0 & a=a} x:=a; y:=1 {y*x!=a! & a>0 & x=a}   y*x!=a! & a>0 & x=a ==> y*x!=a! & x>0
      ------------------------------------------------------------------------------------------------------------ Cons
                              {a>0} x:=a; y:=1 {y*x!=a! & x>0}
```

& : Logisches und

~ : Logisches nicht

Bew. part. Korrektheit: Fakultät (2)

Zweiter Beweis

```
                                    ____                                   ____
                   ---------------------------------- Ass  ------------------------------ Ass
                   {(y*x)*(x-1)!=a! & (x-1)>0} y:=y*x {y*(x-1)!=a! & (x-1)>0}   {y*(x-1)!=a! & (x-1)>0} x:=x-1 {y*x!=a! & x>0}
                   ------------------------------------------------------------------------------- Comp
y*x!=a! & x>0 & x>1 ==> (y*x)*(x-1)!=a! & (x-1)>0   {(y*x)*(x-1)!=a! & (x-1)>0} y:=y*x; x:=x-1 {y*x!=a! & x>0}
---------------------------------------------------------------------------------------- Cons
                   {y*x!=a! & x>0 & x>1} y:=y*x; x:=x-1 {y*x!=a! & x>0}
                   --------------------------------------------------- While
    T'             {y*x!=a! & x>0} WHILE x>1 DO y:=y*x; x:= x-1 OD {y*x!=a! & x>0 & ~(x>1)}   y*x!=a! & x>0 & ~(x>1) ==> y=a!
------------ Cons  ------------------------------------------------------------------------------------------------ Cons
{a>0} x:=a; y:=1 {y*x!=a! & x>0}   {y*x!=a! & x>0} WHILE x>1 DO y:=y*x; x:= x-1 OD {y=a!}
--------------------------------------------------------------------------------------------- Comp
            {a>0} x:=a; y:=1; WHILE x>1 DO y:=y*x; x:=x-1 OD {y=a!}
```

wobei

```
                  ____                                   ____
       ------------------------------ Ass  ------------------------------ Ass
       {1*a!=a! & a>0} x:=a {1*x!=a! & x>0}   {1*x!=a! & x>0} y:=1 {y*x!=a! & x>0}
T' ≡ { ----------------------------------------------------------------- Comp
       a>0 ==> 1*a!=a! & a>0   {1*a!=a! & a>0} x:=a; y:=1 {y*x!=a! & x>0}
       ---------------------------------------------------------------------------- Cons
                        {a>0} x:=a; y:=1 {y*x!=a! & x>0}
```

& : Logisches und

~ : Logisches nicht

Bew. partieller Korrektheit: Division

```
                          ____                                          ____
      Ass ----------------------------------------------  --------------------------------------- Ass
          {x=(q+1)*y+r-y & r-y>=0} q:=q+1 {x=q*y+r-y & r-y>=0}   {x=q*y+r-y & r-y>=0} r:=r-y {x=q*y+r & r>=0}
Comp  ------------------------------------------------------------------------------------------
      (x=q*y+r & r>=0 & r>=y) => (x=(q+1)*y+r-y & r-y>=0)   {x=(q+1)*y+r-y & r-y>=0} q:=q+1; r:=r-y {x=q*y+r & r>=0}
Cons  ------------------------------------------------------------------------------------------
                 {x=q*y+r & r>=0 & r>=y} q:=q+1; r:=r-y {x=q*y+r & r>=0}
While ------------------------------------------------------------------------------------------
      {x=q*y+r & r>=0} WHILE r>=y DO q:=q+1; r:=r-y OD {x=q*y+r & r>=0 & ~(r>=y)}   (x=q*y+r & r>=0 & ~(r>=y)) => (x=q*y+r & r>=0 & r<y)
Cons  ----------------------------------------------------------------------------------------------------------------

        ____                             ____
Ass ------------------------------  ------------------------------ Ass
    {x=0*y+x & x>=0} q:=0 {x=q*y+x & x>=0}  {x=q*y+x & x>=0} r:=x {x=q*y+r & r>=0}
    ---------------------------------------------------------------- Comp
    (x>=0 & y>0) => (x=0*y+x & x>=0) {x=0*y+x & x>=0} q:=0; r:=x {x=q*y+r & r>=0}
    ---------------------------------------------------------------- Cons

    {x>=0 & y>0} q:=0; r:=x {x=q*y+r & r>=0}          {x=q*y+r & r>=0} WHILE r>=y DO q:=q+1; r:=r-y OD {x=q*y+r & r>=0 & r<y}
    ------------------------------------------------------------------------------------------------------------ Comp
               {x>=0 & y>0} q:=0; r:=x; WHILE r>=y DO q:=q+1; r:=r-y OD {x=q*y+r & r>=0 & r<y}
```

& : Logisches und

~ : Logisches nicht

Semantikdefinitionsstile (1)

Es gibt unterschiedliche Stile, die Semantik einer Programmiersprache festzulegen. Sie richten sich an unterschiedliche Adressaten und deren spezifische Sicht auf die Semantik...

Insbesondere unterscheiden wir den...

- *denotationellen*
- *operationellen*
- *axiomatischen*

Stil.

Semantikdefinitionsstile (2)

- *Sprachentwicklersicht*
  - Denotationelle Semantik
- *Sprach- und Anwendungsimplementierersicht*
  - Operationelle Semantik
    - Strukturell operationelle Semantik (small steps semantics)
    - Natürliche Semantik (big steps semantics)
- *Programmierer- und Verifizierersicht*
  - Axiomatische Semantik

Strukturell operationelle Semantik (1)

...noch einmal wiederholt für das Beispiel von WHILE:

$$[\text{skip}_{sos}] \quad \frac{—}{\langle skip,\sigma\rangle \Rightarrow \sigma}$$

$$[\text{abort}_{sos}] \quad \frac{—}{\langle abort,\sigma\rangle \Rightarrow undef}$$

$$[\text{ass}_{sos}] \quad \frac{—}{\langle x{:=}t,\sigma\rangle \Rightarrow \sigma[[\![t]\!]_A(\sigma)/x]}$$

$$[\text{comp}^1_{sos}] \quad \frac{\langle \pi_1,\sigma\rangle \Rightarrow \langle \pi_1',\sigma'\rangle}{\langle \pi_1;\pi_2,\sigma\rangle \Rightarrow \langle \pi_1';\pi_2,\sigma'\rangle}$$

$$[\text{comp}^2_{sos}] \quad \frac{\langle \pi_1,\sigma\rangle \Rightarrow \sigma'}{\langle \pi_1;\pi_2,\sigma\rangle \Rightarrow \langle \pi_2,\sigma'\rangle}$$

Strukturell operationelle Semantik (2)

$$[\text{if}^{tt}_{sos}] \quad \frac{—}{\langle \text{if } b \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi},\sigma\rangle \Rightarrow \langle \pi_1,\sigma\rangle} \qquad [\![b]\!]_B(\sigma) = \texttt{tt}$$

$$[\text{if}^{ff}_{sos}] \quad \frac{—}{\langle \text{if } b \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi},\sigma\rangle \Rightarrow \langle \pi_2,\sigma\rangle} \qquad [\![b]\!]_B(\sigma) = \texttt{ff}$$

$$[\text{while}_{sos}] \quad \frac{—}{\langle \text{while } b \text{ do } \pi \text{ od},\sigma\rangle \Rightarrow \langle \text{if } b \text{ then } \pi;\ \text{while } b \text{ do } \pi \text{ od else } skip \text{ fi},\sigma\rangle}$$

Strukturell operationelle Semantik (3)

Der Fokus liegt auf...

- *individuellen Schritten* einer Berechnungsfolge, d.h. auf der Ausführung von Zuweisungen und Tests

  Intuitive Bedeutung der Transitionsrelation...

  $$\langle \pi, \sigma\rangle \Rightarrow \gamma$$

  ...mit γ von der Form $\langle \pi', \sigma'\rangle$ oder σ' oder $undef$ beschreibt den *ersten* Schritt der Berechnungsfolge von π angesetzt auf σ. Folgende Übergänge sind möglich:

  - γ von der Form $\langle \pi', \sigma'\rangle$:
    Abarbeitung von π nicht vollständig; das Restprogramm π' ist auf σ' anzusetzen
  - γ von der Form σ':
    Abarbeitung von π vollständig; π angesetzt auf σ terminiert in einem Schritt in σ'
  - γ von der Form $undef$:
    Abarbeitung von π terminiert irregulär

Natürliche Semantik (1)

...ebenfalls für das Beispiel von WHILE:

$$[\text{skip}_{ns}] \quad \frac{\text{—}}{\langle skip,\sigma\rangle \to \sigma}$$

$$[\text{abort}_{ns}] \quad \frac{\text{—}}{\langle abort,\sigma\rangle \to undef}$$

$$[\text{ass}_{ns}] \quad \frac{\text{—}}{\langle x{:=}t,\sigma\rangle \to \sigma[[\![\, t \,]\!]_A(\sigma)/x]}$$

$$[\text{comp}_{ns}] \quad \frac{\langle \pi_1,\sigma\rangle \to \sigma', \langle \pi_2,\sigma'\rangle \to \sigma''}{\langle \pi_1;\pi_2,\sigma\rangle \to \sigma''}$$

Natürliche Semantik (2)

$$[\text{if}^{tt}_{ns}] \quad \frac{\langle \pi_1, \sigma\rangle \to \sigma'}{\langle \text{if } b \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi},\sigma\rangle \to \sigma'} \qquad [\![\, b \,]\!]_B(\sigma) = \texttt{tt}$$

$$[\text{if}^{ff}_{ns}] \quad \frac{\langle \pi_2, \sigma\rangle \to \sigma'}{\langle \text{if } b \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi},\sigma\rangle \to \sigma'} \qquad [\![\, b \,]\!]_B(\sigma) = \texttt{ff}$$

$$[\text{while}^{tt}_{ns}] \quad \frac{\langle \pi, \sigma\rangle \to \sigma', \langle \text{while } b \text{ do } \pi \text{ od}, \sigma'\rangle \to \sigma''}{\langle \text{while } b \text{ do } \pi \text{ od},\sigma\rangle \to \sigma''} \qquad [\![\, b \,]\!]_B(\sigma) = \texttt{tt}$$

$$[\text{while}^{ff}_{ns}] \quad \frac{\text{—}}{\langle \text{while } b \text{ do } \pi \text{ od},\sigma\rangle \to \sigma} \qquad [\![\, b \,]\!]_B(\sigma) = \texttt{ff}$$

Natürliche Semantik (3)

Der Fokus liegt auf...

- Zusammenhang von *initialem* und *finalem* Zustand einer Berechnungsfolge

  Intuitive Bedeutung von...

  $$\langle \pi, \sigma\rangle \to \gamma$$

  ...mit γ von der Form σ' oder $undef$ ist: π angesetzt auf initialen Zustand σ terminiert schließlich im finalen Zustand σ' bzw. terminiert irregulär.

Denotationelle Semantik (1)

...auch für das Beispiel von WHILE:

$[\![\, skip \,]\!]_{ds} = id$

$[\![\, abort \,]\!]_{ds} = undef$

$[\![\, x := t \,]\!]_{ds}(\sigma) = \sigma[[\![\, t \,]\!]_A(\sigma)/x]$

$[\![\, \pi_1;\ \pi_2 \,]\!]_{ds} = [\![\, \pi_2 \,]\!]_{ds} \circ [\![\, \pi_1 \,]\!]_{ds}$

$[\![\, \text{if } b \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi} \,]\!]_{ds} = cond([\![\, b \,]\!]_{\mathcal{B}}, [\![\, \pi_1 \,]\!]_{ds}, [\![\, \pi_2 \,]\!]_{ds})$

$[\![\, \text{while } b \text{ do } \pi \text{ od} \,]\!]_{ds} = FIX\ F$

where $F\ g = cond([\![\, b \,]\!]_{\mathcal{B}}, g \circ [\![\, \pi \,]\!]_{ds}, id)$

Denotationelle Semantik (2)

Zur Hilfsfunktion $cond$...

Funktionalität...

$$cond : (\Sigma \to \mathbb{B}) \times (\Sigma \to \Sigma) \times (\Sigma \to \Sigma) \to (\Sigma \to \Sigma)$$

Definiert durch...

$$cond(p, g_1, g_2)\ \sigma =_{df} \begin{cases} g_1\ \sigma & \text{falls } p\ \sigma = \mathtt{tt} \\ g_2\ \sigma & \text{falls } p\ \sigma = \mathtt{ff} \end{cases}$$

Zu den Argumenten und zum Resultat von $cond$...

- 1. Argument: Prädikat (in unserem Szenario total definiert; siehe Vorlesungsteil 1)
- 2.&3. Argument: Je eine partiell definierte Zustandstransformation
- Resultat: Wieder eine partiell definierte Zustandstransformation

Denotationelle Semantik (3)

Zur Hilfsfunktion FIX...

Funktionalität...

$$FIX : ((\Sigma \to \Sigma) \to (\Sigma \to \Sigma)) \to (\Sigma \to \Sigma)$$

Definiert durch...

$$F\ g = cond([\![\, b \,]\!]_{\mathcal{B}}, g \circ [\![\, \Pi \,]\!]_{ds}, id)$$

Daraus ergibt sich...

- FIX ist ein Funktional ("Zustandstransformationsfunktional")
- Die denotationelle Semantik der while-Schleife ist ein Fixpunkt des Funktionals F (und zwar der kleinste!)

Mehr Details zu FIX und Co. später!

Denotationelle Semantik (4)

- *Operationelle* Semantik
  ...der Fokus liegt darauf, *wie* ein Programm ausgeführt wird
- *Denotationelle* Semantik
  ...der Fokus liegt auf dem *Effekt*, den die Ausführung eines Programms hat: Für jedes *syntaktische* Konstrukt gibt es eine *semantische* Funktion, die ersterem ein *mathematisches Objekt* zuweist, i.a. eine Funktion, die den Effekt der Ausführung des Konstrukts beschreibt.

Denotationelle Semantik (5)

Zentral für denotationelle Semantiken: **Kompositionalität**!

Intuitiv:

- Für jedes Element der elementaren syntaktischen Konstrukte/Kategorien gibt es eine zugehörige semantische Funktion
- Für jedes Element eines zusammengesetzten syntaktischen Konstrukts/Kategorie gibt es eine semantische Funktion, die über die semantischen Funktionen der Komponenten des zusammengesetzten Konstrukts definiert ist.

Hauptergebnisse

Theorem

$$\forall\ \pi\ \in\ \textbf{Prg}.\ [\![\,\pi\,]\!]_{sos} = [\![\,\pi\,]\!]_{ns} = [\![\,\pi\,]\!]_{ds}$$